AISG Vulnerability Dossier

AISG-12-000

September 5, 2012

<dtrammell@americaninfosec.com>
http://www.americaninfosec.com/

AISG-12-000 Webmin Privileged Remote Code Execution

Vulnerability Information

Vulnerability Class	Input Validation
Affected Versions Tested	1.580
Affected Versions Assumed	
Unaffected Versions	
Affected Platforms Tested	1: x86-32 Ubuntu Linux 11.10 2: x86-32 Solaris 11.11 3: x86-64 Solaris 11.11 4: x86-32 FreeBSD 9.0
Affected Platforms Assumed	All Vendor-supported Linux All Vendor-supported Solaris All Vendor-supported BSD
Unaffected Platforms	
Reliability Rating	Completely (100%)

Vulnerability Test Matrix

	1	**2**	**3**	**4**
1.580	V	V	V	V

Exploit / Proof-of-Concept Information

Supported Targets	1.580 on x86-32 Linux 1.580 on x86-32 Solaris 11.11 1.580 on x86-64 Solaris 11.11 1.580 on x86-32 FreeBSD 9.0
Attack Vector	Remote
Exploitation Impact	Code Execution*
Exploitation Context	root
Exploitation Indicators	File creation on the filesystem Repeat code execution**
Prerequisites	Successful Authentication
Reliability Rating	Completely (100%)
Development Status	Complete
Development Phase	Metasploit Exploit
Development Goal	Metasploit Exploit
Exploit Features	Triggerable Execution Persistence**

* Successful exploitation allows execution of any perl library or executable residing on the system.
** After successful exploitation, the exploitation trigger and payload remain resident on the system and may be repeatedly triggered.

1 Overview

An input validation flaw allows for authenticated users to execute arbitrary Perl statements, commands, or libraries by parsing any file provided.

2 Impact

Privileged arbitrary code execution as the root user is achievable by leveraging this vulnerability.

3 Technical Explanation

When user input for the CGI variable "type" is passed into */status/save_mon.cgi* it is assigned the name "$serv->{'type'}" and "${type}" in the underlying scripting language, as shown in Code Excerpt 1.

Code Excerpt 1 CGI "type" Variable

```
if ($in{'type'}) {
        $serv->{'type'} = $in{'type'};
```

Later ${type} is reassigned within *statuslib.pl* as "${t}" and used within a filename in a "do" statement without any validation of the user input, as shown in Code Excerpt 2.

Code Excerpt 2 Unvalidated User Input in "do" Statement

```
local $t = $_[0]->{'type'};
...
else {
                do "${t}-monitor.pl" if (!$done_monitor{$t}++);
                local $func = "get_${t}_status";
```

Perl treats null bytes as regular characters whereas the underlying C functions used by Perl to perform the opening of files treat null bytes as terminators. By using a poison null byte it is possible to cause the underlying C functions to open and read an arbitrary file. An example of this would be *index.cgi* reading the data/filename ("/tmp/environ") and additionally passing a null byte at the end of the arbitrary filename. The complete filename as Perl interprets it then becomes "/tmp/environ%00-monitor.pl".

The underlying C functions interpret the null as a terminator and open "/tmp/environ" instead of "/tmp/environ%00-monitor.pl". The data from that file is then passed into the Perl interpreter and inserted into a "do" statement.

save_mon.cgi causes the arbitrary filename to be saved into configuration variables under *$webminroot/etc/status/services/<epochtime>.serv*, as shown in Code Excerpt 3.

Code Excerpt 3 save_mon.cgi Configuration Variables

```
runon=0
depend=
ontimeout=
remote=\*
email=
ondown=
clone=
onup=
tmpl=
fails=1
desc=Alive System
groups=
type=/tmp/environ\^@
id=1331832761
notify=snmp sms pager
nosched=0
```

This file is then parsed by */status/index.cgi* which utilizes the *service_table* method as shown in Code Excerpt 4 to read all service files from */etc/webmin/status/services* and subsequently calls the *service_status* method within the *status-lib.pl* library. The filename information is parsed and passed into the *service_status* method as ${t}. The variable ${t} is passed into a "do" statement within *status-lib.pl*, as shown in Code Excerpt 5.

Code Excerpt 4 /status/index.cgi: service_table Function

```
        if ($config{'index_status'}) {
                @stats = &service_status($s, 1);
```

Code Excerpt 5 status-lib.pl: service_status Function

```
do "${t}-monitor.pl" if (!$done_monitor{$t}++);
```

In Perl, "do" can be passed a block or group of statements to be parsed or a subroutine; however, it may also be passed a filename. When passed a filename such as "do 'filename.pl' " the underlying Perl interpreter treats it as though the filename had been passed to an *eval()* method.

Therefore, because the arbitrary data is being assigned to variable "$t" and passed as part of a filename within a "do" statement without any input validation it is possible to insert arbitrary data into that filename. This allows an attacker to tell the Perl interpreter to open and *eval()* an arbitrary file. For example, when *index.cgi* parses the "type" variable from the saved configuration file the "do" statement may become as shown in Code Excerpt 6. This is equivalent to the statement "eval '/tmp/environ'' and causes all lines in */tmp/environ* to be interpreted and executed by the Perl interpreter.

Code Excerpt 6 index.cgi Example

```
do "/tmp/environ%00-monitor.pl";
```

It should be noted that the same vulnerability with variable "$type" exists within *save_mon.cgi*; however, directory traversal (appending one or more '../'s) must be utilized to exploit the vulnerability in that location.